LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

ECT CODE: 310248

**LAB MANUAL**

**Laboratory Practice-I (System Programming & Operating System)**

**Semester – I, Academic Year: 2021-22**

1

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

**310248: Laboratory Practice-I**

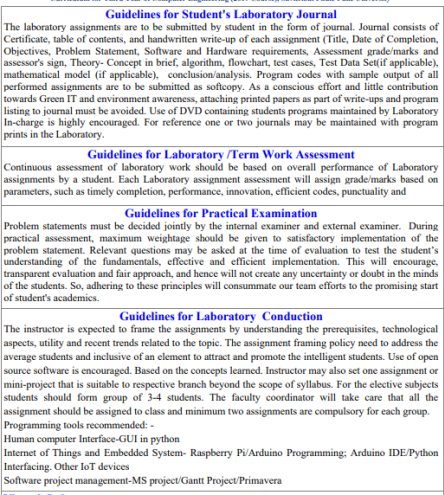
**Teaching Scheme: Examination Scheme:** Practical: 4 Hrs/Week Term work: 25 Marks Credits: 02 Practical: 25 Marks **List of Laboratory Assignments**

|  |  |  |
| --- | --- | --- |
| **Sr.**  **No.** | **Group A** | **Page No.** |
| 1 | Design suitable data structures and implement pass-I of a two-pass assembler for pseudo-machine in Java using object oriented feature. Implementation should consist of a few instructions from each category and few assembler directives. | 5 |
| 2 | Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment. | 9 |
| 3 | Design suitable data structures and implement pass-I of a two-pass macro processor using OOP features in Java | 13 |
| 4 | Write a Java program for pass-II of a two-pass macro-processor. The output of assignment-3 (MNT, MDT and file without any macro definitions) should be input for this assignment. | 16 |
| 5 | Write a program to create Dynamic Link Library for any mathematical operation and write an application program to test it. (Java Native Interface / Use VB or VC++). | 19 |
|  | **Group B** |  |
| 6 | Write a program to solve Classical Problems of Synchronization using Mutex and Semaphore. | 22 |
| 7 | Write a program to simulate CPU scheduling algorithms: FCFS , SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive) | 25 |
| 8 | Write a program to simulate memory replacement strategies- First Fit, Best Fit, Worst Fit and Nest Fit. | 28 |
| 9 | Write a program to simulate page replacement algorithms using 1. FIFO 2. Least Recently Used (LRU) 3.Optimal algorithm | 31 |

2

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

Instructions

3

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

**Course Objectives:**

1. To learn system programming tools

2. To learn modern operating system

**Course Outcome:**

On completion of this course, learners will be able to

CO1: Implement different system software’s like assembler, macro processor, DLL, etc. CO2: Implement concept of synchronization and concurrency

CO3: Implement scheduling policies and memory management concepts of operating system

4

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Assignment No.: 01** |
| **Problem Statement:** Design suitable data structures and implement pass-I of a two-pass assembler for pseudo machine in Java/C++ using object oriented feature. Implementation should consist of a few instructions from each category and few assembler directives. |
| **Objectives:**  1. To study the design and implementation of 1st pass of two pass assembler. 2. To study the categorized instruction set of assembler.  3. To study the data structure used in assembler implementation. |
| **Theory:**  1. Explain various Data and Instruction formats of assembly language programming. 2. Explain the design of Pass- I of assembler with the help of flowchart and example. 3. Discuss various Data structure used in Pass-I along with its format and significance of each field. |
| **Algorithm/Flowchart:** |
| **Design diagrams (if any):**  1. Class Diagram  2. Use case Diagram  3. ER Diagram |
| **Input:**  Source code of Assembly Language  SAMPLE START 100  USING \*, 15  L 1, FOUR |

5

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| A 1, =F’3’  ST 1, RESULT  SR 1, 2  LTORG  L 2, FIVE  A 2, =F’5’  A 2, =F’7’  FIVE DC F’5’  FOUR DC F’4’  RESULT DS 1F  END |

**Output:**

100 SAMPLE START 100

100 USING \*, 15

100 L 1, FOUR

104 A 1, =F’3’

108 ST 1, RESULT

112 SR 1, 2

114 LTORG

124 L 2, FIVE

128 A 2, =F’5’

132 A 2, =F’7’

136 FIVE DC F’5’

140 FOUR DC F’4’

144 RESULT DS 1F

152 5

156 7

160 END

**Machine Opcode Table (MOT)**

|  |  |  |  |
| --- | --- | --- | --- |
| Mnemonic | Hex / Binary  Code | Length (Bytes) | Format |
| L | 5A | 4 | RX |
| A | 1B | 4 | RX |
| ST | 50 | 4 | RX |
| SR | 18 | 2 | RR |

**Pseudo Opcode Table (POT)**

|  |  |
| --- | --- |
| Pseudo op | Address / Name of Procedure to implement pseudo operation |
| START | PSTART |
| USING | PUSING |
| DC | PDC |
| DS | PDS |
| LTORG | PLTORG |
| END | PEND |

6

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

**Symbol Table (ST)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. No | Symbol name | Address | Value | Length | Relocation |
| 1 | SAMPLE | 100 | -- | 160 | R |
| 2 | FIVE | 136 | 5 | 4 | R |
| 3 | FOUR | 140 | 4 | 4 | R |
| 4 | RESULT | 144 | \_\_ | 4 | R |

**Literal Table (LT)**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No | Literal | Address | Length |
| 1 | 3 | 120 | 4 |
| 2 | 5 | 152 | 4 |
| 3 | 7 | 156 | 4 |

|  |
| --- |
| **Instructions :**  Not specific |
| **Test Cases:**  1. Check syntax of instruction (Correct and wrong)  2. Symbol not found  3. Wrong instruction  4. Duplicate symbol declaration  5. Test the output of program by changing value of START pseudo opcode. 6. Test the output of program by changing position of LTORG pseudo-op. |
| **Software Requirement:**  1. Fedora  2. Eclipse  3. JDK |
| **Hardware Requirement:**  Not specific |
| **Frequently Asked Questions:**  1. What is two pass assembler?  2. What is the significance of symbol table?  3. Explain the assembler directives EQU, ORIGIN.  4. Explain the assembler directives START, END, LTORG.  5. What is the use of POOLTAB and LITTAB?  6. How literals are handled in pass I?  7. What are the tasks done in Pass I?  8. How error handling is done in pass I?  9. Which intermediate data structures are designed and implemented in PassI? 10. What is the format of a machine code generated in PassII?  11. What is forward reference? How it is resolved by assembler?  12. How error handling is done in pass II?  13. What is the difference between IS, DL and AD? |

|  |
| --- |
| 14. What are the tasks done in Pass II? |
| **Conclusion:**  Input assembly language program is processed by applying Pass-I algorithm of assembler and intermediate data structures, Symbol Table, Literal Table, MOT, POT, BT, etc. are generated. |

8

|  |
| --- |
| **Assignment No.: 02** |
| **Problem Statement:**  Implement Pass-II of two pass assembler for pseudo-machine in Java/C++ using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment. |
| **Objectives:**  1. To study the design and implementation of 2nd pass of two pass assembler. 2. To study the data structure used in Pass-2 of assembler implementation. |
| **Theory:**  1. Explain the design of Pass- II of assembler with the help of flowchart and example. |
| **Algorithm/Flowchart:** |
| **Design diagrams (if any):**  1. Class Diagram  2. Use case Diagram  3. ER Diagram |
| **Input:**  Intermediate code of pass-1.  **LC LABEL INSTR. OPERANDS** |

9

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

**----------------------------------------------------------------**

100 SAMPLE START 100

100 USING \*, 15

100 L 1, FOUR

104 A 1, =F’3’

108 ST 1, RESULT

112 SR 1, 2

114 LTORG

124 L 2, FIVE

128 A 2, =F’5’

132 A 2, =F’7’

136 FIVE DC F’5’

140 FOUR DC F’4’

144 RESULT DS 1F

152 5

156 7

160 END

**Machine Opcode Table (MOT)**

|  |  |  |  |
| --- | --- | --- | --- |
| Mnemonic | Hex / Binary  Code | Length (Bytes) | Format |
| L | 5A | 4 | RX |
| A | 1B | 4 | RX |
| ST | 50 | 4 | RX |
| SR | 18 | 2 | RR |

**Pseudo Opcode Table (POT)**

|  |  |
| --- | --- |
| Pseudo op | Address / Name of Procedure to implement pseudo operation |
| START | PSTART |
| USING | PUSING |
| DC | PDC |
| DS | PDS |
| LTORG | PLTORG |
| END | PEND |

**Symbol Table (ST)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. No | Symbol name | Address | Value | Length | Relocation |
| 1 | SAMPLE | 100 | -- | 160 | R |
| 2 | FIVE | 136 | 5 | 4 | R |
| 3 | FOUR | 140 | 4 | 4 | R |
| 4 | RESULT | 144 | \_\_ | 4 | R |

10

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Literal Table (LT)**  Sr. No Literal Address Length  1 3 120 4  2 5 152 4  3 7 156 4 |
| **Output:**  **Base Table (BT)**  Register no Availability Value/ Contents  1 N --  : : :  : : :  : : :  15 Y 100  **Object Code**  **LC OPCODE OPERAND**  ------------------------------------------------------------  100 5A 1,40(0,15)  104 1B 1,20(0,15)  108 50 1,44(0,15)  112 18 1,2  124 5A 2,36(0,15)  128 1B 2,52(0,15)  132 1B 2,56(0,15) |
| **Instructions :**  **1.**  **2.**  **3.** |
| **Test Cases:**  1. Check syntax of instruction (Correct and wrong)  2. Symbol not found  3. Wrong instruction  4. Duplicate symbol declaration  5. Test the output of program by changing value of START & USING pseudo opcode. |
| **Software Requirement:**  1. Fedora  2. Eclipse  3. JDK |
| **Hardware Requirement:** |

11

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Frequently Asked Questions:**  1. What is two pass assembler?  2. What is the significance of symbol table?  3. Explain the assembler directives EQU, ORIGIN.  4. Explain the assembler directives START, END, LTORG.  5. What is the use of POOLTAB and LITTAB?  6. How literals are handled in pass I?  7. What are the tasks done in Pass I?  8. How error handling is done in pass I?  9. Which intermediate data structures are designed and implemented in PassI? 10. What is the format of a machine code generated in PassII?  11. What is forward reference? How it is resolved by assembler?  12. How error handling is done in pass II?  13. What is the difference between IS, DL and AD? |
| **Conclusion:**  The intermediate data structures generated in Pass-I of assembler are given as input to the Pass-II of assembler, processed by applying Pass-II algorithm of assembler and machine code is generated |

12

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Assignment No.: 03** |
| **Problem Statement:** Design suitable data structures and implement Pass-I of a two pass macro processor using OOP features in Java/C++. The output of Pass-I (MNT, MDT, ALA & Intermediate code file without any macro definitions) should be input for Pass-II. |
| **Objectives:**  1. To identify and design different data structure used in macro-processor implementation  2. To apply knowledge in implementation of two pass microprocessor. |
| **Theory:**  1. What is macro processor?  2. Differentiate Macro and Function?  3. Explain the design of Pass- I of macro-processor with the help of flowchart? 4. Explain the design of Data structure used in Pass-I?  5. Explain the data structures used in Pass-I? |
| **Algorithm/Flowchart:** |

13

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
|  |
| **Design diagrams (if any):**  1. Class diagram  2. Sequence diagram  3. |
| **Input:**  Small assembly language program with macros written in file input.asm.  MACRO  &lab ADDS &arg1,&arg2  &lab L 1, &arg1  A 1, &arg2  MEND  PROG START 0  BALR 15,0  USING \*,15  LAB ADDS DATA1, DATA2  ST 4,1  DATA1 DC F’3’  DATA2 DC F’4’  END |
| **Output:**  Assembly language program without macro definition but with macro call.  **Note:** Follow the following templates during implementation  **Macro Name Table (MNT) :**  **Macro Definition Table (MDT) :**  **Argument List Array (ALA) :** |
| **Instructions :**  **1.**   |  | | --- | | **Assignment No.: 04** | | **Problem Statement:** Design suitable data structures and implement Pass-II of a two pass macro processor using OOP features in Java/C++. The output of Pass-I (MNT, MDT, ALA & Intermediate code file without any macro definitions) should be input for Pass-II. | | **Objectives:**  1. To identify and design different data structure used in macro-processor implementation 2. To apply knowledge in implementation of pass-2 of two pass microprocessor. | | **Theory:**  1. Explain design steps of two pass microprocessor, types of statements, data structures required and flowcharts. | | **Algorithm/Flowchart:** | | **Design diagrams (if any):**  1. Class diagram  2. Sequence diagram  3. | | **Input:** Output of pass-1 (Intermediate File) given as a input to pass-2. |   **2.**  **3.** |
| **Test Cases:** |

|  |
| --- |
| **1.** Check macro end not found.  **2.** Duplicate macro name found.  **3.** Check program output by changing macro name and parameter list.  **4.** Handle label in macro definition.  **5.** Handle multiple macro definitions and calls |
| **Software Requirement:**  **1.** Fedora  **2.** Eclipse  **3.** JDK |
| **Hardware Requirement: N/A** |
| **Frequently Asked Questions:**  **1.** Define macro?  **2.** Define purpose of pass-1 of two pass macro processor  **3.** List out types of macro arguments  **4.** What is the use of MDT-index field in MNT?  **5.** What we store in ALA? |
| **Conclusion:** We have successfully completed implementation of Pass-I of macro processor. |

16

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| PROG START 0  BALR 15,0  USING \*,15  LAB ADDS DATA1, DATA2  ST 4,1  DATA1 DC F’3’  DATA2 DC F’4’  END |
| **Output:**  Assembly language program without macro definition and macro call.  PROG START 0  BALR 15,0  USING \*,15  LAB L 1, DATA1  A 1, DATA2  ST 4,1  DATA1 DC F’3’  DATA2 DC F’4’  END |
| **Instructions :**  **1.**  **2.**  **3.** |
| **Test Cases:** |

17

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| 1. Check macro definition not found.  2. Check program output by changing parameter list in macro call. |
| **Software Requirement:**  **1.** Fedora  **2.** Eclipse  **3.** JDK |
| **Hardware Requirement: N/A** |
| **Frequently Asked Questions:**  **1.** What is macro expansion?  **2.** Define purpose of pass-2 of two pass macro processor  **3.** What is positional arguments?  **4.** What is the use of MDT-index field in MNT?  **5.** What is the use of MNT table while processing macro call? |
| **Conclusion:** We have successfully completed implementation of Pass-II of macro processor. |

18

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Assignment No.: 05** |
| **Problem Statement:**  Write a program to create a Dynamic Link Library for any mathematical operations (arithmetic, trigonometric and string operation) and write an application program to test it. (Java Native Interface/Use VB/VC++) |
| **Objectives:**  1. To study and understand concept of DLL  2. To understand JNI  3. To implement DLL using JNI |
| **Theory:**  1. What is DLL? Significance of DLL. Advantages/ Disadvantages of DLL 2. What is Native Interface? Reasons to use JNI.  3. What is shared object? |
| **Algorithm/Flowchart:**  1. **Write a Java Class that uses C Codes - TestJNI.java**  public class TestJNI {  static {  System.loadLibrary(“cal"); // Load native library at runtime  // cal.dll (Windows) or libcal.so (Unix)  }  // Declare a native method add() that receives nothing and returns void private native int add (int n1,int n2);  // Test Driver  public static void main(String[] args) {  // invoke the native method  System.out.println(“Addition is=”+new TestJNI().add(10,20);  } }  *Compile Java code:*  javac TestJNI.java  2. **Create the C/C++ Header file - TestJNI.h**  javah -jni TestJNI  3. **C Implementation - TestJNI.c**  #include <jni.h>  #include <stdio.h>  #include “TestJNI.h"  // Implementation of native method add() of TestJNI class |

19

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| JNIEXPORT jint JNICALL Java\_TestJNI\_add(JNIEnv \*env, jobject thisObj,jint n1,jint n2) {  jint res;  res=n1+n2;  return res;  }  *Compile c-program:*  $gcc -I /usr/local/jdk1.8.0\_91/include /usr/local/jdk1.8.0\_91/include/linux -o libcal.so –shared TestJNI.c  **4. Run java program**  $java -Djava.library.path=. TestJNI  Addition is=30  5. **Repeat step 1-4 for all mathematical operations mentioned in problem statement. Flowchart:** |
| **Design diagrams (if any):**  1. Use Case Diagram  2. Sequence Diagram |
| **Input:** |

20

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| 1. n1=20  2. n2=10 |
| **Output:**  1. Addition=30 |
| **Instructions :**  1. This assignment can be implemented using VB application and C++ DLL using visual studio on windows |
| **Test Cases:**  1. Divide by zero  2. Missing arguments |
| **Software Requirement:**  1. Fedora  2. Jdk  3. Eclipse/ equivalent IDE |
| **Hardware Requirement:** |
| **Frequently Asked Questions:**  1. Difference between static link library and dynamic link library  2. What is shared object?  3. Advantages/Disadvantages of using JNI |
| **Conclusion:**  Successfully implemented DLL and tested it with java application |

21

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Assignment No.: 06** |
| **Problem Statement:**  Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non Preemptive) and Round Robin (Preemptive). |
| **Objectives:**  1. To study the process management and various scheduling policies viz. Preemptive and Non preemptive.  2. To study and analyze different scheduling algorithms. |
| **Theory :**  1. Define process. Explain need of process scheduling.  2. Explain different scheduling criteria and policies for scheduling processes. 3. Explain possible process states  4. Explain FCFS, SJF(Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive) and determine waiting time, turnaround time, throughput using each algorithm. |
| **Algorithm/Flowchart:**  1. **FCFS**  1. Input the processes along with their burst time (bt).  2. Find waiting time (wt) for all processes.  3. As first process that comes need not to wait so waiting time for process 1 will be 0 i.e. wt[0] = 0.  4. Find **waiting time** for all other processes i.e. for process i ->  wt[i] = bt[i-1] + wt[i-1] .  5. Find **turnaround time** = waiting\_time + burst\_time for all processes.  6. Find **average waiting time** = total\_waiting\_time / no\_of\_processes.  7. Similarly, find **average turnaround time** = total\_turn\_around\_time / no\_of\_processes.  1. **SJF**  1. Traverse until all process gets completely executed.  a) Find process with minimum remaining time at every single time lap.  b) Reduce its time by 1.  c) Check if its remaining time becomes 0  d) Increment the counter of process completion.  e) Completion time of current process = current\_time +1;  e) Calculate waiting time for each completed process.  wt[i]= Completion time - arrival\_time-burst\_time  f) Increment time lap by one. |

22

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| 2. Find turnaround time (waiting\_time+burst\_time).  3. **Priority**  1- First input the processes with their burst time and priority.  2- Sort the processes, burst time and priority according to the priority.  3- Now simply apply FCFS algorithm.  4. **RR**  1- Create an array **rem\_bt[]** to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array)  2- Create another array **wt[]** to store waiting times of processes. Initialize this array as 0. 3- Initialize time : t = 0  4- Keep traversing the all processes while all processes are not done. Do following for i'th process if it is not done yet.  a- If rem\_bt[i] > quantum  (i) t = t + quantum  (ii) bt\_rem[i] -= quantum;  c- Else // Last cycle for this process  (i) t = t + bt\_rem[i];  (ii) wt[i] = t - bt[i]  (ii) bt\_rem[i] = 0; // This process is over |
| **Design diagrams (if any):**  Class Diagram  Use Case Diagram  Sequence Diagram |
| **Input:**  1. Enter the number of processes  2. Enter burst time and arrival time of each process |
| **Output:**  1. Compute Waiting time, turnaround time, average waiting time, average turnaround time and throughput.  For each algorithm display result as follows:  Process Burst Time Arrival  Waiting Time Turnaround  Time  Time  P1  P2  P3  -  Calculate |

a. b. c.

23

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| 1. Average waiting time=  2. Average turnaround time=  3. Throughput= |
| **Instructions :**  **1.**  **2.**  **3.** |
| **Test Cases:**  1. Check arrival time of all process should not be same. |
| **Software Requirement:**  1. Fedora  2. Eclipse  3. JDK |
| **Hardware Requirement: for simulation no dependency** |
| **Frequently Asked Questions:**  1. What are the types of CPU scheduler?  2. What is the difference between long and short term scheduling?  3. Logic of program?  4. What is preemptive and non-preemptive scheduling?  5. What are types of scheduling algorithms?  6. Why Priority scheduling may cause low-priority processes to starve?  7. What are the goals of scheduling?  8. Define the difference between preemptive and non-preemptive scheduling. 9. Which scheduling algorithm is best? Why? |
| **Conclusion:**  CPU policies implemented successfully |

24

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Assignment No.: 7** |
| **Problem Statement:**  Write a program to solve classical problems of synchronization using mutex and semaphore. |
| **Objectives:**  1. To understand reader writer synchronization problem  2. To solve reader-writer synchronization problem using mutex and semaphore |
| **Theory:**  • There is a data area shared among a number of processor registers.  • The data area could be a file, a block of main memory, or even a bank of processor registers.  • There are a number of processes that only read the data area (readers) and a number that only write to the data area (writers).  • The conditions that must be satisfied are  ⮚ Any number of readers may read simultaneously read the file.  ⮚ Only one write at a time may write to the file.  ⮚ If a writer is writing to the file, no reader may read it.  **Semaphore:**  **Definition:** Semaphores are system variables used for synchronization of process **Two types of Semaphore:**  ⮚ **Counting semaphore –** integer value can range over an unrestricted domain ⮚ **Binary semaphore –**  ∙ Integer value can range only between 0 and 1; can be simpler to implement ∙ Also known as mutex locks  **Semaphore functions:**  **Package: import java.util.concurrent.Semaphore;**  1) **To initialize a semaphore:**  **Semaphore Sem1 = new Semaphore(1);**  2) **To wait on a semaphore:**  **/\* Wait (S)**  **while S<=0**  **no-op;**  **S - -;**  **\*/**  **Sem1.acquire();**  3) **To signal on a semaphore:**  **/\* Signal(S)**  **S ++;**  **\*/**  **mutex.release();** |

25

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Algorithm/Flowchart:**  **Algorithm for Reader Writer:**  **1. import java.util.concurrent.Semaphore;**  **2. Create a class RW**  **3. Declare semaphores – mutex and wrt**  **4. Declare integer variable readcount = 0**  **5. Create a nested class Reader implements Runnable**  **a. Override run method (Reader Logic)**  **i.** wait(mutex);  **ii.** readcount := readcount +1;  **iii.** if readcount = 1 then  **iv.** wait(wrt);  **v.** signal(mutex);  **vi.** …  **vii.** reading is performed  **viii.** …  **ix.** wait(mutex);  **x.** readcount := readcount – 1;  **xi.** if readcount = 0 then signal(wrt);  **xii.** signal(mutex):  **6. Create a nested class Writer implements Runnable**  **a. Override run method (Writer Logic)**  **i.** wait(wrt);  **ii.** …  **iii.** writing is performed  **iv.** …  **v.** signal(wrt);  **7. Create a class main**  **a. Create Threads for Reader and Writer**  **b. Start these thread** |
| **Design diagrams (if any):** |
| **Input:**  1. Number of Readers  2. Number of Writers |
| **Output:**  1. Execution of Readers and Writers |
| **Instructions:**  1.  2.  3. |
| **Test Cases:** |

26

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| 1. Create 5 readers first and then 5 writers and check their sequence of execution 2. Create 5 writers first and then 5 readers and check their sequence of execution 3. Create 5 writers and 5 readers alternatively and check their sequence of execution |
| **Software Requirement:**  1. Java  2. Eclipse/NetBeans |
| **Hardware Requirement:**  1. Nothing Special |
| **Frequently Asked Questions:**  1. What is synchronization of threads?  2. Explain reader writer problem  3. Explain wait and sequence functions  4. What is semaphore.  5. What are different types of semaphore |
| **Conclusion:** Implemented Reader Writer synchronization problem using semaphores in Java |

27

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Assignment No.: 08** |
| **Problem Statement:**  Write a Java/C++ program to simulate memory placement strategies  1. First Fit  2. Best Fit  3. Worst Fit  4. Next Fit |
| **Objectives:**  1. To acquire knowledge memory placement strategies  2. To be able to implement memory placement strategies |
| **Theory:**  1. Why we need memory placement strategies?  2. What is fragmentation?  3. Explain working of memory placement strategies with suitable example. |
| **Algorithm/Flowchart:**  **1. First Fit algorithm/pseudo code**  o Read all required input  o FOR i<-0 to all jobs ‘js’  ▪ FOR j<-0 to all blocks ‘bs’  o IF block[j]>=jobs[i]  ▪ Check jth block is already in use or free  ∙ Continue and search next free block  ▪ Otherwise allocate jth block to ith job  o Display all job with allocated blocks and fragmentation  **2. First Fit algorithm/pseudo code**  o Read all required input  o FOR i<-0 to all jobs ‘js’  ▪ SET BestInd🡨 -1  ▪ FOR j<-0 to all blocks ‘bs’  o IF block[j]>=jobs[i]  ▪ IF Block is free and BestInd==-1 THEN SET  BestInd🡨j  ▪ ELSEIF Block is free and block[BestInd]>block[j]  THEN SET BestInd🡨j  ▪ ELSE continue with next block  ∙ Continue and search next free block |

28

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| ▪ IF BestInd!=-1 THEN allocate jth block to ith job  o Display all job with allocated blocks and fragmentation  **3. Worst Fit Algorithm/Pseudo code**  o Read all required input  o FOR i<-0 to all jobs ‘js’  ▪ SET WstInd🡨 -1  ▪ FOR j<-0 to all blocks ‘bs’  o IF block[j]>=jobs[i]  ▪ IF Block is free and WstInd==-1 THEN SET  WstInd🡨j  ▪ ELSEIF Block is free and block[WstInd]<block[j]  THEN SET WstInd🡨j  ▪ ELSE continue with next block  ∙ Continue and search next free block  ▪ IF WstInd!=-1 THEN allocate jth block to ith job  o Display all job with allocated blocks and fragmentation  4. As above write algorithm of Next Fit strategies |
| **Design diagrams (if any):**  1. Class diagram |
| **Input:**  • No. of jobs (js) & No. of blocks (bs)  • Job size of all jobs & Block size of all blocks  For Example:  js=4  bs=5  block[] = {100, 500, 200, 300, 600};  jobs[] = {212, 417, 112, 426}; |
| **Output:**  **Sample output of Worst Fit algorithm (same way generate o/p for other algorithms)-**  Process No. Process Size Block no.  1 212 5  2 417 2 |

29

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| 3 112 4  4 426 Not Allocated |
| **Instructions : not specific** |
| **Test Cases:** |
| **Software Requirement:**  1. Eclipse IDE  2. Java |
| **Hardware Requirement: Not specific** |
| **Frequently Asked Questions:**  1. Which algorithm is best and why?  2. Need of allocating blocks to jobs?  3. What is the time taken by each algorithm for execution? |
| **Conclusion:** successfully implemented simulation of memory placement strategies. |

30

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **Assignment No.: 09** |
| **Problem Statement:**  Write a Java Program (using OOP features) to implement paging simulation using 1. FIFO  2. Least Recently Used (LRU)  3. Optimal algorithm |
| **Objectives:**  1. To study page replacement policies to understand memory management. 2. To understand efficient frame management using replacement policies. |
| **Theory:**  **CONCEPT OF PAGE REPLACEMENT:**  1. Page Fault: Absence of page when referenced in main memory during paging leads to a page fault.  2. Page Replacement: Replacement of already existing page from main memory by the required new page is called as page replacement. And the techniques used for it are called as page replacement algorithms.  **NEED OF PAGE REPLACEMENT:**  Page replacement is used primarily for the virtual memory management because in virtual memory paging system principal issue is replacement i.e. which page is to be removed so as to bring in the new page, thus the use of the page replacement algorithms. Demand paging is the technique used to increase system throughput. To implement demand paging page replacement is primary requirement. If a system has better page replacement technique it improves demand paging which in turn drastically yields system performance gains.  **PAGE REPLACEMENT POLICIES:**  1. Determine which page to be removed from main memory.  2. Find a free frame.  1) If a frame is found use it  2) if no free frame found, use page replacement algorithm to select a victim frame.  3) Write the victim page to the disk.  3. Read the desired page into the new free frame, change the page and frame tables. 4. Restart the user process. |

31

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **PAGE REPLACEMENT ALGORITHMS:**  **1. FIFO**  This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.  2. **OPTIMAL PAGE REPLACEMENT ALGORITHM:** Replace the page that will not be used for longest period of time as compared to the other pages in main memory. An optimal page replacement algorithm has lowest page fault rate of all algorithm. It is called as OPT or MIN.  **ADVANTAGE:**  1) This algorithm guarantees the lowest possible page-fault rate for a fixed no. of frames.  **DISADVANTAGE:**  1) The optimal page replacement algorithm is very difficult to implement, as it requires the knowledge of reference strings i.e. strings of memory references. 3. **LEAST RECENTLY USED (LRU):** LRU algorithm uses the time of the page’s last usage. It uses the recent past as an approximation of the near future, then we can replace the page that has not been used for the longest period of the time i.e. the page having larger idle time is replaced.  **ADVANTAGE:**  1) The LRU policy is often used for page replacement and is considered to be good.  **DISADVANTAGES:**  1) It is very difficult to implement.  2) Requires substantial hardware assistance.  3) The problematic determination of the order for the frames defined by the time of last usage |
| **Algorithm/Flowchart:**  **1. FIFO :**  **1.** Start the process  **2.** Read number of pages n |

32

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| **3.** Read number of pages no  **4.** Read page numbers into an array a[i]  **5.** Initialize avail[i]=0 .to check page hit  **6.** Replace the page with circular queue, while re-placing check page availability in the frame Place avail[i]=1 if page is placed in the frame Count page faults  **7.** Print the results.  **8.** Stop the process.  2. **LEAST RECENTLY USED**  1.Start the process  **2.** Declare the size  **3.** Get the number of pages to be inserted  **4.** Get the value  **5.** Declare counter and stack  **6.** Select the least recently used page by counter value  **7.** Stack them according the selection.  **8.** Display the values  **9.** Stop the process  3. **OPTIMAL**  **ALGORTHIM:**  **1.** Start Program  **2.** Read Number Of Pages And Frames  3.Read Each Page Value  **4.** Search For Page In The Frames  **5.** If Not Available Allocate Free Frame  **6.** If No Frames Is Free Replace The Page With The Page That Is Least Used 7.Print Page Number Of Page Faults  8.Stop process. |
| **Design diagrams (if any):**  1. Class Diagram |
| **Input:**  1. Number of frames  2. Number of pages |

33

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU

|  |
| --- |
| 3. Page sequence |
| **Output:**  1. Sequence of allocation of pages in frames (for each algorithm)  2. Cache hit and cache miss ratio. |
| **Instructions :** |
| **Test Cases:**  1. Test the page hit and miss ratio for different size of page frames.  2. Test the page hit and miss ratio for both algorithms with different page sequences. |
| **Software Requirement:**  1. Fedora  2. Eclipse  3. JDK |
| **Hardware Requirement:** |
| **Frequently Asked Questions:**  1. What is virtual memory?  2. Explain working of LRU page replacement algorithm  3. Explain working of OPTIMAL page replacement algorithm  4. Which Page replacement algorithm is best?  5. Explain what is Belody’s Anomaly?  6. Explain the scenario in which page replacement algorithm is used?  7. Explain what is page fault?  8. Explain what is paging scheme?  9. Explain what is counting based page replacement algorithms? |
| **Conclusion:** Successfully implemented all page replacement policies. |

34

LP-1 (SPOS Part) 2019 Course Lab Manual | SPPU 35